

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Carol Ann Egan et al.

Serial No.: 10/665,656

Filed: September 18, 2003

For: COMPUTER APPLICATION AND METHODS FOR
AUTONOMIC UPGRADE MAINTENANCE OF COMPUTER
HARDWARE, OPERATING SYSTEMS AND APPLICATION
SOFTWARE

Group Art Unit: 2192

Confirmation No.: 8562

**AMENDED APPEAL BRIEF IN SUPPORT OF APPEAL FROM
THE PRIMARY EXAMINER TO THE BOARD OF APPEALS**

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Applicants hereby submit an Amended Appeal Brief in response to the Notification of Non-Compliant Appeal Brief dated April 22, 2009. An appeal brief was previously filed March 30, 2009, but was rejected as noncompliant with 37 CFR 41.37. The present amended appeal brief is being filed within the 30-day time period allowed to file a compliant appeal brief.

1. Real Party in Interest

International Business Machines Corporation is the real party in interest.

2. Related Appeals and Interferences

There are no related appeals or interferences pending with this application.

3. Status of Claims

Appellants appeal from the rejection in the November 14, 2008 Final Office Action of claims 1-34 and 36-54. Claim 35 has been cancelled. The claims on appeal are set forth in Appendix A.

4. Status of Amendments

An Amendment-After-Final to the final rejection of November 14, 2008 was filed on January 13, 2009. In an Advisory Action dated 2/18/2009, the Examiner indicated that, for purposes of appeal, the proposed amendments will be entered.

5. Summary of Claimed Subject Matter

As described in the specification at page 3, paragraph 7, the present invention provides methods and a computer-readable program for providing autonomic, event driven upgrade maintenance of one or more software modules residing on a computer system. In a preferred embodiment, a method begins by detecting a predefined triggering event on the computer system indicative of a potential maintenance issue. Next, the computer system connects to an upgrade management server, where the upgrade maintenance server creates a list of recommended upgrade modules based upon the

triggering event and a set of selection policies to download to the computer system. The method then downloads the list of recommended upgrade modules from the upgrade management server to the computer system, and selectively installs upgrade modules chosen from the list of upgrade modules on the computer system. The user is then notified of the status of the upgrade maintenance operation.

Appellants are appealing from the Examiner's rejection of claims 1-34 and 36-54. Claim 1 is an independent claim. Claims 2-8, 16-19, and 28-29 depend directly from claim 1. Claims 9-15 depend directly from claim 8. Claims 20-27 depend directly from claim 19. Claim 30 is an independent claim. Claims 31-34, 36-37, and 45 depend directly from claim 30. Claim 35 has been cancelled. Claims 38-44 depend directly from claim 37. Claims 46-53 depend directly from claim 45. Claim 54 is an independent claim.

In compliance with 37 C.F.R. § 41.37c(1)(v), a concise explanation of the subject matter defined in independent claims 1, 30, and 54, including references to the specification by page and line number, and to the drawings follow.

Claim 1 describes a method for providing autonomic, event-driven upgrade maintenance of one or more software modules residing on a computer system (Specification, page 3, paragraph 7, lines 1-3, and Fig. 4). The method begins by detecting a predefined triggering event on the computer system indicative of a potential maintenance issue, the predefined triggering event being triggered by a current operating condition of the computer system (Specification, page 14, lines 1-3, page 7, paragraph 20, and Fig. 4, element 304). Next, the method connects to an upgrade management server based upon a set of user defined policies residing on the computer system (Specification, page 14, paragraph 34, lines 3-5 and Fig. 4, element 306). The method then creates on the upgrade management server a list of recommended upgrade modules to download to

the computer system, wherein the list is based upon a set of selection policies (Specification, page 14, paragraph 34, lines 5-7 and Fig. 4, element 308). Next, the method downloads the list of recommended upgrade modules from the upgrade management server to the computer system (Specification, page 14, lines 7-10 and Fig. 4, element 310). Finally, the method selectively installs upgrade modules chosen from the list of recommended upgrade modules downloaded to the computer system, based upon the set of user defined policies residing on the computer system (Specification, page 14, paragraph 34, lines 10-11 and Fig. 4, element 312).

Claim 30 provides a computer-readable program stored on a computer-readable medium (Specification, page 8, paragraph 24, and Fig. 4). The program begins by detecting a predefined triggering event on the computer system indicative of a potential maintenance issue, the predefined triggering event being triggered by a current operating condition of the computer system (Specification, page 14, lines 1-3, page 7, paragraph 20, and Fig. 4, element 304). Next, the program connects to an upgrade management server based upon a set of user defined policies (Specification, page 14, paragraph 34, lines 3-5 and Fig. 4, element 306). The program then creates on the upgrade management server a list of recommended upgrade modules to download to the computer system, wherein the list is based upon a set of selection policies (Specification, page 14, paragraph 34, lines 5-7 and Fig. 4, element 308). Next, the program downloads the list of recommended upgrade modules from the upgrade management server to the computer system (Specification, page 14, lines 7-10 and Fig. 4, element 310). Finally, the program selectively installs upgrade modules chosen from the list of recommended upgrade modules downloaded to the computer system, based upon the set of user defined policies residing on the computer system (Specification, page 14, paragraph 34, lines 10-11 and Fig. 4, element 312).

Claim 54 provides a method for deploying computing infrastructure, comprising integrating computer-readable code into a computing system, wherein the code in combination with the computing system is capable of providing autonomic, event-driven upgrade maintenance of one or more software modules residing on a computer system (Specification, page 8, paragraph 24, and Figs 1 and 4). The method begins by detecting a predefined triggering event on the computer system indicative of a potential maintenance issue, the predefined triggering event being triggered by a current operating condition of the computer system (Specification, page 14, lines 1-3, page 7, paragraph 20, and Fig. 4, element 304). Next, the method connects to an upgrade management server, based upon a set of user defined policies (Specification, page 14, paragraph 34, lines 3-5 and Fig. 4, element 306). The method then creates on the upgrade management server a list of recommended upgrade modules to download to the computer system, wherein the list is based upon a set of selection policies (Specification, page 14, paragraph 34, lines 5-7 and Fig. 4, element 308). Next, the method downloads the list of recommended upgrade modules from the upgrade management server to the computer system (Specification, page 14, lines 7-10 and Fig. 4, element 310). Finally, the method selectively installs upgrade modules chosen from the list of recommended upgrade modules downloaded to the computer system, based upon the set of user defined policies residing on the computer system (Specification, page 14, paragraph 34, lines 10-11 and Fig. 4, element 312).

6. Grounds of Rejection to be Reviewed on Appeal

The Examiner has objected to claim 33 because of a minor informality. Appellants respectfully submit that this objection was previously addressed by Appellants in an Amendment After Final dated January 13, 2009, and in an advisory action dated 2/18/2009, the Examiner indicated that, for purposes of appeal, the proposed amendments will be entered. Thus, Appellants respectfully submit the objection to claim 33 is in error.

The Examiner has rejected claims 1-34 and 36-54 under 35 U.S.C. § 103(a) as being unpatentable over “SafePatch Version 0.9 Manual,” March 1999 (art made of record, hereafter “SafePatch”) in view of Chamberlain, U.S. Patent 7,073,172 B2 (art made of record, hereafter “Chamberlain”). The issue is whether the Examiner is correct in asserting that claims 1-34 and 36-54 are obvious under 35 U.S.C. § 103 over SafePatch in view of Chamberlain.

7. Argument

Rejection under 35 U.S.C. § 103(a)

Independent claims 1, 30 and 54 contain the claim element, “detecting a predefined triggering event on the computer system indicative of a potential maintenance issue, the predefined triggering event being triggered by a current operating condition of the computer system” (i.e., the “first claim element”). Appellants respectfully submit that this claim element is neither disclosed nor suggested, alone or in combination, by the “SafePatch” and/or “Chamberlain” references.

SafePatch version 0.9 provides automated analysis of network-based computer systems to determine the status of security patches and distributes needed patches. SafePatch determines what patches need to be installed and what patches are installed on a system. SafePatch will distribute patches to the remote system for later installation. For those patches that are installed, SafePatch checks the permissions and ownership of the files referenced in the patch and reports on the attributes that differ from those recommended by the patch (SafePatch Version 0.9 User Manual, page 1, paragraph 1). The SafePatch reference is relied upon by the Examiner in his argument regarding the claim element of “detecting a predefined triggering event on the computer system

indicative of a potential maintenance issue, the predefined triggering event being triggered by a current operating condition of the computer system”.

Chamberlain (US Patent 7,073,172) describes a method and mechanism for automatically patching software implementations such as application as they are installed from an external source. A patch program is run for a software implementation that is advertised as available, or is otherwise available, though not yet installed (or fully installed). The mechanism maintains the patch information for that software implementation. Whenever the software implementation is installed, the mechanism determines that the software implementation has the patch information maintained therefore. The mechanism automatically applies the patch as part of the on-demand installation process. The patch may change the files, the values of registry entries and/or the installation logic associated with the software implementation.

Appellants submit that the Examiner only applies Chamberlain with regard to the claim element of “selectively installing upgrade modules chosen from the set of recommended upgrade modules downloaded to the computer system, based upon the set of user defined policies residing on the computer system” (i.e., the “fifth” claim element) of claims 1, 30 and 54. The fifth claim element is not in dispute in the current appeal. Further, Appellants have reviewed the Chamberlain reference with regard to the first and second claim elements of claims 1, 30 and 54, and submits that Chamberlain neither discloses nor suggests these elements. More specifically, with regard to the first claim element, Chamberlain does not provide a “predefined triggering event”... “indicative of a potential maintenance issue”. With regard to the second claim element, Chamberlain does not provide an “upgrade management server” which is connected to based on a “set of user policies”.

On page 3, paragraph 2 of the November 14, 2008 Final Office Action, the Examiner states that SafePatch provides the first claim element of “detecting a predefined triggering event on the computer system indicative of a potential maintenance issue, the predefined triggering event being triggered by a current operating condition of the computer system” at: 1) page 2, section 1.1.2; 2) page 1, SafePatch Overview; and 3) page 43, steps 1-3 and related text.

With regard to the first cited passage, page 2, section 1.1.2, paragraph 4 of SafePatch states that “The SafePatch administrator controls the evaluation of remote systems through the part of the SafePatch Server called the Patch Server. ... The time, date and how often a job is to occur can be specified for each remote system or for a group of systems”. Thus, in the SafePatch reference, there is no “predefined, triggering event indicative of a maintenance, the predefined triggering event being triggered by a current operating condition of the computer system” occurring on the remote server. Rather, **an administrator controls the evaluation** from the upgrade management server (i.e., SafePatch server) connected to the remote system to be upgraded, **specifying a date and time** when the evaluation occurs. Similarly, the SafePatch Overview on page 1 describes a system wherein the upgrade management (i.e., SafePatch) server remotely queries the remote system (i.e., the system to receive the upgrades) on a scheduled basis. Finally the flow diagram on page 43, step 1-3 once again describes a system wherein the upgrade management (i.e., SafePatch) server initiates the update process, rather than having the update process being initiated at the remote server (i.e., the server receiving the upgrade) via an event trigger, as described in the present invention.

Appellants respectfully submit that the “SafePatch” reference actually teaches away from the present invention, since: 1) the evaluation of remote systems in the SafePatch system is triggered at the **upgrade management server** (i.e., the SafePatch server), not at the **remote system receiving the upgrade**, as claimed by the present

invention; and 2) the evaluation is driven by the upgrade management server by an administrator on a **periodic, scheduled time basis**, rather than a **“predefined, triggering event indicative of a maintenance issue”** occurring on the remote server itself, as claimed by the present invention.

This “event driven” “bottoms up” approach of the present invention offers several advantages over the conventional “time driven”, “top-down” approach taught by the SafePatch reference. First, the “time-driven”, “top-down” approach of SafePatch generally will consume much more computer resource than the “event driven” “bottoms-up” approach of the present invention, since SafePatch will require a plurality of periodic, scheduled checks from the upgrade management server to the remote computer system that will likely result in no updates being required in most instances. By contrast, the present invention’s approach will only occur asynchronously (and likely much less frequently) only when an actual triggering event occurs at the remote system. Also, the approach of the present invention does not require an administrator at the upgrade management server to set up a periodic scheme for checking the remote system, since the remote system will tell the upgrade management server itself when it needs an update via the triggering event generated at the remote system.

Since the SafePatch reference neither discloses nor suggests first claim element of independent claims 1, 30 and 54, Appellants respectfully once again respectfully submit that claims 1, 30, and 54 are now in condition for allowance.

Claims 2-29, 31-34 and 36-53 are dependent claims which rely, either directly or indirectly, from independent claims 1, and 30. Appellants submit that since claims 1 and 30 are in condition for allowance for reasons stated above, claims 2-29, 31-34 and 36-53 are also now in condition for allowance.

The second claim element of independent claims 1, 30 and 54 involves “connecting to an upgrade management server, based upon a set of user defined policies residing on the computer system”. The Examiner states that the “SmartPatch” reference provides this claim element at pp. 17-20, SafePatch Server; and page 43, steps 3-5 and related text. Appellants respectfully traverse this rejection.

Appellants respectfully submit that the selection policies employed by the “SmartPatch” reference occur at the upgrade management server, not at the computer system receiving the update. The set of user defined policies for the present invention is defined in the specification at page 9, paragraph 27. Examples of user defined policies include the a preferred connection time for connecting to the upgrade management server 104, scheduled day/time for auto applying upgrades to the computer system 102, a specific defined time to connect to the upgrade management server to check for updates, etc. Thus, the “SmartPatch” reference actually teaches away from the present invention, since the “SmartPatch” reference drives the upgrade management from the upgrade management server on a periodic basis and with human intervention (see “SmartPatch, page 2, section 1.1.1.2 “The SafePatch administrator controls the evaluation of remote systems through the part of the SafePatch Server called the Patch Server. ... The time, date and how often a job is to occur can be specified for each remote system or a group of systems”). In contrast, the present invention drives the upgrade maintenance operation from the remote system on an event triggered basis (see Specification, page 9, paragraph 26), and subject to policies defined on the remote system to be updated, not to policies defined by an administrator on the upgrade management server. Simply stated, “SmartPatch” defines a top-down approach to upgrade management, wherein the upgrader system drives the maintenance operation, while the present invention describes a bottoms-up approach to upgrade management, wherein the upgrade system drives and controls the maintenance operation.

In the Advisory Action of February 18, 2009, the Examiner states “SafePatch explicitly teaches: pages 17-20, selecting/adding remote computer and installing SafePatch Agent (i.e., SafePatch Agent has “user defined policies residing on the computer system”); steps 3-5, Figure in page 43, SafePatch Agent installed and running on the remote computer, collecting file attributes and reporting them to SafePatch server, but not to any upgrade server (i.e., “connecting to an upgrade management server, based upon a set of user defined policies residing on the computer system”).

Appellants respectfully submit that the Examiner mischaracterizes the passages from the SafePatch reference stated above. First, Appellants are confused by the Examiner’s use of the term SafePatch “Agent” since the term “Agent” does not occur anywhere within pages 17-20. Thus, Appellants have no clue what the Examiner refers to when he refers to “selecting/adding remote computers and installing SafePatch Agent”.

When referring to the Figure on page 43, the Examiner once again refers to a mysterious “SafePatch Agent” running on the remote computer, yet the term “SafePatch Agent” is nowhere to be found in either the figure or the accompanying text. Similarly, Appellants do not understand how steps 3-5 teach “connecting to an upgrade management server, based upon a set of user defined policies residing on the computer system”.

Step 3 of Figure on page 43 states “compare data on remote systems with patch data”, step 4 states “determine file and patch status”, and step 5 states “Check ACL and owner setting of INSTALLED patches”. None of these steps neither disclose nor suggest “connecting to an upgrade management server, based upon a set of user defined policies residing on the computer system” as claimed by the Examiner. In fact, the Examiner does not mention Step 1 of the Figure on page 43, which states “collect patches for this system”, and is described in the accompanying text as “the SafePatch Server connects to the remote system to determine its operating system type”. Thus, it is the SafePatch

Server (i.e., the upgrade management server), not the remote computer which initiates the process.

Since the SafePatch reference neither discloses nor suggests the second claim element of independent claims 1, 30 and 54, Appellants respectfully submit that claims 1, 30 and 54 are now in condition for allowance.

8. Claims Appendix

1. (Previously Amended) A method for providing autonomic, event-driven upgrade maintenance of one or more software modules residing on a computer system, the method comprising:

detecting a predefined triggering event on the computer system indicative of a potential maintenance issue, the predefined triggering event being triggered by a current operating condition of the computer system;

connecting to an upgrade management server, based upon a set of user defined policies residing on the computer system;

creating on the upgrade management server a list of recommended upgrade modules to download to the computer system, the list based upon a set of selection policies;

downloading a set of recommended upgrade modules from the upgrade management server to the computer system; and

selectively installing upgrade modules chosen from the set of recommended upgrade modules downloaded to the computer system, based upon the set of user defined policies residing on the computer system.

2. (Original) The method of claim 1, wherein the method further comprises the step of:

notifying a user of the status of the upgrade maintenance operation.

3. (Original) The method of claim 1, wherein the predefined triggering event comprises a change to the hardware configuration of the computer system.
4. (Original) The method of claim 1, wherein the predefined triggering event comprises a change to the software configuration of the computer system.
5. (Original) The method of claim 1, wherein the predefined triggering event comprises exceeding a predefined error threshold on the computer system.
6. (Original) The method of claim 1, wherein the predefined triggering event comprises exceeding a predefined performance threshold on the computer system.
7. (Original) The method of claim 1, wherein the predefined triggering event comprises exceeding a predefined elapsed time since the last connection to the upgrade management server.
8. (Original) The method of claim 1, wherein the steps of connecting to a upgrade management server and selectively installing the list of recommended upgrade modules are controlled by a set of user defined policies.
9. (Original) The method of claim 8, wherein the set of user defined policies includes a preferred connection time.
10. (Original) The method of claim 8, wherein the set of user defined policies includes the connection resource to be used.

11. (Original) The method of claim 8, wherein the set of user defined policies includes the specification of computer system areas/software products to enable automatic application of upgrades.
12. (Original) The method of claim 8, wherein the set of user defined policies includes a defined time to connect to the upgrade management server to check for upgrades.
13. (Original) The method of claim 8, wherein the set of user defined policies includes a defined elapsed time interval for connecting to the upgrade management server to check for upgrades.
14. (Original) The method of claim 8, wherein the set of user defined policies includes a notification list for e-mailing user of information and actions relative to the upgrade management process.
15. (Original) The method of claim 8, wherein the set of user defined policies include a list of one or more upgrade management servers to be used for the upgrade management process.
16. (Previously Amended) The method of claim 1, wherein the one or more computer software modules comprises software applications.
17. (Previously Amended) The method of claim 1, wherein, the one or more computer-software modules comprises operating systems.
18. (Previously Amended) The method of claim 1, wherein the one or more computer software modules comprises device drivers for installed hardware components.

19. (Original) The method of claim 1, wherein the set of selection policies is sent from the computer system to the upgrade management server.
20. (Original) The method of claim 19, wherein the set of selection policies includes creating the list of recommended upgrade modules based upon a specific set of upgrades requested by the computer system.
21. (Previously Amended) The method of claim 19, wherein the set of selection policies includes comparing a revision levels of the one or more software modules residing on the computer system against a revision levels of one or more software modules residing on the upgrade management server.
22. (Original) The method of claim 19, wherein the set of selection policies includes creating the list of recommended upgrade modules by identifying modules associated with a hardware change on the computer system.
23. (Original) The method of claim 19, wherein the set of selection policies includes creating the list of recommended upgrade modules by identifying software modules associated with a software change on the computer system.
24. (Original) The method of claim 19, wherein the set of selection policies includes creating the list of recommended upgrade modules by identifying upgrades specifically associated with an error triggering event on the computer system.
25. (Original) The method of claim 19, wherein the set of selection policies includes creating the list of recommended upgrade modules by identifying upgrades specifically associated with a performance triggering event on the computer system.

26. (Original) The method of claim 19, wherein the set of selection policies includes creating the list of recommended upgrade modules by analyzing a problem history provided by the computer system.

27. (Original) The method of claim 19, wherein the set of selection policies includes creating the list of recommended upgrade modules by identifying compatible revision levels between two or more software modules included within the list of modules.

28. (Original) The method of claim 1, wherein the step of downloading the list of recommended upgrade modules from the upgrade management server to the computer system further comprises the step of downloading the upgrade modules themselves from the upgrade management server to the computer system.

29. (Original) The method of claim 1, wherein the step of selectively installing upgrade modules chosen from the list of recommended upgrade modules on the computer system further comprises the step of downloading any upgrade modules chosen from the list of recommended upgrade modules from the upgrade management server to the computer system prior to the install.

30. (Previously Amended) A computer-readable program stored on a computer-readable storage medium, said computer readable program being configured to perform the steps of:

detecting a predefined triggering event on a computer system indicative of a potential maintenance issue, the predefined triggering event being triggered by a current operating condition of the computer system;

connecting to an upgrade management server, based upon a set of user defined policies residing on the computer system;

creating on the upgrade management server a list of recommended upgrade modules to download to the computer system, the list based upon a set of selection policies;

downloading a set of recommended upgrade modules from the upgrade management server to the computer system; and

selectively installing upgrade modules chosen from the set of recommended upgrade modules downloaded to the computer system, based upon the set of user defined policies residing on the computer system.

31. (Original) The computer-readable program of claim 30, wherein the computer-readable program further includes the step of:

notifying a user of the status of the upgrade maintenance operation.

32. (Original) The computer-readable program of claim 30, wherein the predefined triggering event comprises a change to the hardware configuration of the computer system.
33. (Previously Amended) The computer-readable program of claim 30, wherein the predefined triggering event comprises a change to the software configuration of the computer system.
34. (Original) The computer-readable program of claim 30, wherein the predefined triggering event comprises exceeding a predefined error threshold on the computer system.
35. (Cancelled)
36. (Original) The computer-readable program of claim 30, wherein the predefined triggering event comprises exceeding a predefined elapsed time since the last connection to the upgrade management server.
37. (Original) The computer-readable program of claim 30, wherein the steps of connecting to a upgrade management server and selectively installing the list of recommended upgrade modules are controlled by a set of user defined policies.
38. (Original) The computer-readable program of claim 37, wherein the set of user defined policies includes a preferred connection time.
39. (Original) The computer-readable program of claim 37, wherein the set of user defined policies includes the connection resource to be used.

40. (Original) The computer-readable program of claim 37, wherein the set of user defined policies includes the specification of computer system areas/software products to enable automatic application of upgrades.

41. (Original) The computer-readable program of claim 37, wherein the set of user defined policies includes a defined time to connect to the upgrade management server to check for upgrades.

42. (Original) The computer-readable program of claim 37, wherein the set of user defined policies includes a defined elapsed time interval for connecting to the upgrade management server to check for upgrades.

43. (Original) The computer-readable program of claim 37, wherein the set of user defined policies includes a notification list for e-mailing user of information and actions relative to the upgrade management process.

44. (Original) The computer-readable program of claim 37, wherein the set of user defined policies include a list of one or more upgrade management servers to be used for the upgrade management process.

45. (Original) The computer-readable program of claim 30, wherein the set of selection policies is sent from the computer system to the upgrade management server.

46. (Original) The computer-readable program of claim 45, wherein the set of selection policies includes creating the list of recommended upgrade modules based upon a specific set of upgrades requested by the computer system.

47. (Previously Amended) The computer-readable program of claim 45, wherein the set of selection policies includes comparing a revision level of the one or more software modules residing on the computer system against a revision level of one or more software modules residing on the upgrade management server.

48. (Original) The computer-readable program of claim 45, wherein the set of selection policies includes creating the list of recommended upgrade modules by identifying modules associated with a hardware change on the computer system.

49. (Original) The computer-readable program of claim 45, wherein the set of selection policies includes creating the list of recommended upgrade modules by identifying software modules associated with a software change on the computer system.

50. (Original) The computer-readable program of claim 45, wherein the set of selection policies includes creating the list of recommended upgrade modules by identifying upgrades specifically associated with an error triggering event on the computer system.

51. (Original) The computer-readable program of claim 45, wherein the set of selection policies includes creating the list of recommended upgrade modules by identifying upgrades specifically associated with a performance triggering event on the computer system.

52. (Original) The computer-readable program of claim 45, wherein the set of selection policies includes creating the list of recommended upgrade modules by analyzing a problem history provided by the computer system.

53. (Original) The computer-readable program of claim 45, wherein the set of selection policies includes creating the list of recommended upgrade modules by identifying compatible revision levels between two or more software modules included within the list of modules.

54. (Previously Amended) A method for deploying computing infrastructure, comprising integrating computer-readable code into a computing system, wherein the code in combination with the computing system is capable of providing autonomic, event-driven upgrade maintenance of one or more software modules residing on a computer system, the method comprising the steps of:

detecting a predefined triggering event on a computer system indicative of a potential maintenance issue, the predefined triggering event being triggered by a current operating condition of the computer system;

connecting to an upgrade management server, based upon a set of user defined policies residing on the computer system;

creating on the upgrade management server a list of recommended upgrade modules to download to the computer system, the list based upon a set of selection policies;

downloading a set of recommended upgrade modules from the upgrade management server to the computer system; and

selectively installing upgrade modules chosen from the set of recommended upgrade modules downloaded to the computer system, based upon the set of user defined policies residing on the computer system.

9. Evidence Appendix

There is no evidence attached for this appeal.

10. Related Proceedings Appendix

There are no related proceedings. Therefore, there are no copies of decisions rendered by a court of the Board attached here.

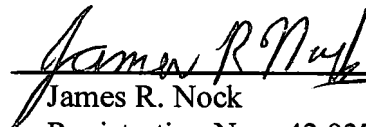
Appellants believe this appendix satisfies the requirements of 37 C.F.R. § 41.37(c)(x).

Respectfully submitted,

Date: May 12, 2009

Telephone: (507) 253-4661
Fax No.: (507) 253-2382

By: _____


James R. Nock
Registration No.: 42,937
IBM Corporation - Department 917
3605 Highway 52 North
Rochester, Minnesota 55901-7829